PoT-PTQ: A Two-step Power-of-Two Post-training for LLMs

Xinyu Wang [1] **Vahid Partovi Nia** [2] **Peng Lu** [3] **Jerry Huang** [3 4] **Xiao-Wen Chang** [1] **Boxing Chen** [2] **Yufei Cui** [2]

Abstract

Large Language Models (LLMs) are powerful but resource-intensive. Power-of-two (PoT) quantization offers hardware-friendly compression but often struggles with accuracy, especially on GPUs due to sign bit entanglement and sequential dequantization. We propose PoT-PTQ, a novel PoT quantization framework for LLM weights that achieves state-of-the-art accuracy in extremely low-precision (2- and 3-bit) and enables faster inference via efficient dequantization. Our two-step post-training algorithm initializes quantization scales robustly and refines them with a minimal calibration set. PoT-PTQ surpasses integer quantization baselines at low precisions and achieves dequantization speedups of up to $3.67\times$ on NVIDIA V100 and $1.63\times$ on NVIDIA RTX 4090 compared to uniform integer dequantization.

1. Introduction

Large Language Models (LLMs) (Brown et al., 2020; Zhang et al., 2022; Team et al., 2023; Yang et al., 2024) have demonstrated remarkable capabilities but their extensive computational and memory footprints hinder widespread deployment, particularly on resource-constrained edge devices. Quantization, which reduces numerical precision, is a critical technique for compressing LLMs. Methods are broadly categorized into Quantization-Aware Training (QAT), which requires costly retraining, and Post-Training Quantization (PTQ), which adapts pretrained models with minimal calibration data (Nagel et al., 2021). PTQ can target weights only or both weights and activations. While weight-only PTQ for formats like INT8 or INT4 (Dettmers et al., 2022a; Lin et al., 2024) has shown promise, performance often degrades significantly at ultra-low bit-widths (e.g., 2-bit or 3-bit). Furthermore, inference typically involves dequantizing weights to FP16 for GEMM operations, adding latency.

Power-of-Two (PoT) quantization, where values are restricted to $\pm 2^E$, is an attractive alternative. It offers inherent hardware efficiency as multiplications can be replaced by bit shifts (You et al., 2020; Elhoushi et al., 2021). Statistically, PoT levels, being logarithmically spaced, align well with the bell-shaped or exponential distributions of weights in trained LLMs (Figure 3 in Appendix), potentially offering better approximation than uniform quantization (Li et al., 2023). However, prior PoT methods for deep learning (Li et al., 2020; 2021), when applied to LLMs, often suffer severe accuracy loss. This is due to the coarse, non-linear nature of PoT levels and the difficulty of finding optimal scaling factors, especially under aggressive compression. Naive dequantization of PoT values on GPUs can also be inefficient due to bit-level dependencies.

To address these limitations, we propose PoT-PTQ (**PoT** **P**ost-**T**raining **Q**uantization), a novel framework for LLMs. Our main contributions are:

- A specialized two-stage PTQ algorithm that first robustly initializes PoT scales data-agnostically, then refines them with a lightweight, data-dependent calibration tailored to PoT's structure.
- State-of-the-art accuracy for LLM quantization at 2-bit and 3-bit precision, outperforming strong integer PTQ baselines like GPTQ (Frantar et al., 2023) and OmniQuant (Shao et al., 2024).
- A highly optimized GPU dequantization kernel for PoT values that leverages bitwise parallelism, achieving significant inference speedups.

2. Preliminaries: PoT Quantization

We focus on weight-only PoT quantization for transformer-based LLMs. Each full-precision weight W_{ij} in layer l is quantized to $\widetilde{W}_{ij}^{(l)} = S_{ij}^{(l)} \cdot P_{ij}^{(l)} \cdot 2^{E_{ij}^{(l)}}$. Here, $S_{ij}^{(l)} \in \mathbb{R}_+$ is the learnable scaling factor, $P_{ij}^{(l)} = \text{sign}(W_{ij}^{(l)})$ is the sign bit, and $E_{ij}^{(l)} \in \mathbb{N}_0$ is the quantized exponent. $E_{ij}^{(l)}$ is

[1]McGill University [2]Noah's Ark Lab [3]Université de Montréal [4]Mila - Quebec AI Institute. Correspondence to: Xinyu Wang <xinyu.wang5@mail.mcgill.ca>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

computed as:

$$E_{ij}^{(l)} = \text{clamp}\left(\text{round}\left(\log_2\left(|W_{ij}^{(l)}|/S_{ij}^{(l)}\right)\right), 0, q_{\max}\right), \tag{1}$$

where $q_{\max} = 2^{n-1} - 1$ for an n-bit quantization scheme (excluding the sign bit, which is stored separately), and $\text{clamp}(x, a, b) = \min(\max(x, a), b)$. To balance model compression and accuracy, we employ **group-wise quantization**. Each column of the weight matrix $\mathbf{W}^{(l)}$ is partitioned into fixed-size groups (e.g., $G = 64$ or 128 contiguous rows). All weights within a group share the same scaling factor $S_{ij}^{(l)}$. This significantly reduces the overhead of storing scales while allowing adaptability to local weight distributions.

The logarithmic spacing of PoT levels provides finer resolution for small magnitude weights, which are abundant in LLMs, compared to the uniform spacing of integer quantization (Figure 3, Appendix). This statistical alignment is a key motivation for PoT. However, the discrete and non-linear nature of PoT makes scale optimization challenging. Existing PTQ methods, often designed for smoother uniform quantization error landscapes, struggle to find effective PoT scales, leading to substantial accuracy loss. Our POT-PTQ framework is specifically engineered to overcome these PoT-specific optimization hurdles.

3. Two-Step Power-of-Two Post-Training Quantization (POT-PTQ)

Our POT-PTQ framework, illustrated in Figure 1, uses a two-step algorithm to optimize PoT quantization scales for LLM weights.

3.1. Step 1: Data-Agnostic Scale Initialization

The primary difficulty in PoT quantization is the highly non-smooth error surface with respect to the scaling factors $S^{(l)}$. This is due to the $\text{round}(\log_2(\cdot))$ operation in Equation (1), where small changes in $S_{ij}^{(l)}$ can cause discrete jumps in the assigned exponent $E_{ij}^{(l)}$, leading to abrupt changes in reconstruction error (Figure 4, middle, in Appendix). This makes gradient-based optimization of scales from scratch unstable and ineffective. Moreover, as shown in Figure 4 (right), the optimal scales often differ substantially from naive choices (e.g., $b = 1$).

Step 1 addresses this by initializing group-wise scales s^* using only the weight statistics, without any calibration data. For each weight group $\mathbf{W}_{\text{group}}$ (a contiguous subvector of G weights from a column of $\mathbf{W}^{(l)}$), we seek a scale multiplier b^* that minimizes the L2 reconstruction error:

$$b^* = \arg\min_{b \in \mathcal{B}} \left\| \mathbf{W}_{\text{group}} - \widetilde{\mathbf{W}}_{\text{group}}(b) \right\|_2^2, \tag{2}$$

where the reconstructed group is $\widetilde{\mathbf{W}}_{\text{group}}(b) = (s_0 \cdot b) \cdot \mathbf{P}_{\text{group}} \circ 2^{\mathsf{E}_{\text{group}}(b)}$. $\mathbf{P}_{\text{group}} = \text{sign}(\mathbf{W}_{\text{group}})$ stores the signs, and $\mathsf{E}_{\text{group}}(b)$ are the exponents calculated using the candidate scale $s_0 \cdot b$ as per Equation (1). The base scale s_0 is set to $s_0 = \frac{\max|\mathbf{W}_{\text{group}}|}{2^{q_{\max}-1}}$, which aligns the largest magnitude in the group with the maximum PoT level $2^{q_{\max}-1}$ if $b = 1$. We perform a grid search for b over a discrete set of multipliers $\mathcal{B} = \{0.01 \cdot i \mid i = 1, \ldots, 200\}$. This range for b (0.01 to 2.00) allows sufficient flexibility to adjust s_0. The optimal scale for the group is $s^* = s_0 \cdot b^*$. This search is performed independently and can be fully parallelized across all groups in $\mathbf{W}^{(l)}$, yielding the initial scale matrix $\mathbf{S}^{(l)}$. Alg. 1 (Appendix) details this process.

3.2. Step 2: Data-Dependent Fine-Tuning

The scales $\mathbf{S}^{(l)}$ from Step 1 are optimized for weight reconstruction error but may not be optimal for preserving the layer's output function due to complex interactions with input activations $\mathbf{X}$. Step 2 refines these scales using a small calibration dataset. This is a lightweight fine-tuning stage that learns a low-dimensional residual adjustment $\mathbf{\Gamma}$ for the scales, avoiding costly full model retraining.

Let $\mathbf{X}$ be the input hidden states to layer l. The layer's operation (e.g., a linear transformation in attention or MLP blocks) is denoted by $\mathcal{F}^{(l)}(\mathbf{W}^{(l)}, \mathbf{X})$. We aim to minimize the Frobenius norm of the output difference:

$$\min_{\mathbf{\Gamma}} Q_2(\mathbf{\Gamma}) = \left\| \mathcal{F}^{(l)}(\mathbf{W}^{(l)}, \mathbf{X}) - \mathcal{F}^{(l)}(\widetilde{\mathbf{W}}^{(l)}(\mathbf{\Gamma}), \mathbf{X}) \right\|_F^2 + \frac{\lambda}{2}\|\mathbf{\Gamma}\|_F^2, \tag{3}$$

where λ is an L2 regularization hyperparameter for $\mathbf{\Gamma}$. The refined group-wise scale $\hat{\mathbf{S}}_{ij}^{(l)}(\mathbf{\Gamma})$ is parameterized as:

$$\hat{\mathbf{S}}_{ij}^{(l)}(\mathbf{\Gamma}) = \mathbf{S}_{ij}^{(l)} \cdot (1 + \mathbf{\Gamma}_{ij}), \tag{4}$$

where $\mathbf{S}_{ij}^{(l)}$ are the robust scales from Step 1. The learnable parameter $\mathbf{\Gamma}_{ij}$ shares the same group structure as the scales (i.e., one Γ value per group) and is initialized to zero. The dequantization process then uses these refined scales $\hat{\mathbf{S}}^{(l)}(\mathbf{\Gamma})$ to compute exponents $\mathsf{E}_{ij}^{(l)}(\mathbf{\Gamma})$ and reconstruct weights $\widetilde{\mathbf{W}}_{ij}^{(l)}(\mathbf{\Gamma})$. The rounding in Equation (1) is non-differentiable. We use the Straight-Through Estimator (STE) (Bengio et al., 2013) for the gradient of $\mathsf{E}_{ij}^{(l)}$ with respect to $\hat{\mathbf{S}}_{ij}^{(l)}$: $\frac{\partial \mathsf{E}_{ij}^{(l)}}{\partial \hat{\mathbf{S}}_{ij}^{(l)}} \approx \frac{\partial}{\partial \hat{\mathbf{S}}_{ij}^{(l)}} \log_2\left(\frac{|\mathbf{W}_{ij}^{(l)}|}{\hat{\mathbf{S}}_{ij}^{(l)}}\right) = -\frac{1}{\hat{\mathbf{S}}_{ij}^{(l)} \ln 2}$. This allows end-to-end backpropagation to optimize $\mathbf{\Gamma}$. This step is highly efficient, learning only one scalar per weight group (e.g., for a group size of 128, this is $128\times$ fewer parameters than learning individual weight adjustments). It typically converges in a few epochs with a small calibration set (e.g., 128 sequences). Alg. 2 (Appendix) provides the procedure.

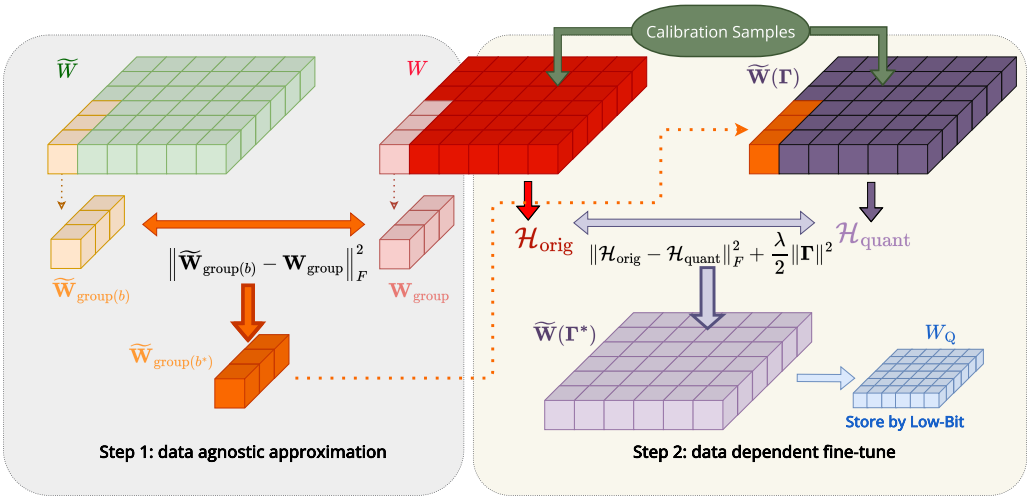

Figure 1: The two-step PoT-PTQ algorithm. **Step 1**: Data-agnostic scale initialization adjusts scales by aligning weight matrices to minimize reconstruction error per group. **Step 2**: Data-dependent fine-tuning refines these scales by aligning layer outputs using a minimal calibration dataset, learning a small residual adjustment.

4. Efficient Dequantization for PoT

A significant advantage of PoT quantization is its potential for highly efficient dequantization, which is critical for accelerating inference, especially the GEMM operations that dominate LLM computation. Conventional uniform quantization reconstructs weights as $\widetilde{\mathbf{W}}^{(l)} = (\mathbf{W}_{\mathrm{Q}}^{(l)} - \mathbf{Z}) \circ \mathbf{S}_{\text{uniform}}$, where $\mathbf{W}_{\mathrm{Q}}^{(l)}$ are integer quantized values, $\mathbf{Z}$ is an integer zero-point, and $\mathbf{S}_{\text{uniform}}$ is a floating-point scale. Dequantization involves floating-point subtraction and multiplication, which can be a bottleneck.

In our PoT scheme, weights are reconstructed as $\widetilde{\mathbf{W}}^{(l)} = \mathbf{S}^{(l)} \circ (-1)^{\mathsf{S}_{\mathsf{bit}}} \circ 2^{\mathsf{E}_{\mathsf{val}}}$. Here, $\mathbf{S}^{(l)}$ are the FP16 scales from our two-step optimization, $\mathsf{S}_{\mathsf{bit}}$ is the stored sign bit, and $\mathsf{E}_{\mathsf{val}}$ is the integer exponent value derived from the n-bit quantized representation. The key insight is that the multiplication by $2^{\mathsf{E}_{\mathsf{val}}}$ can be implemented by directly manipulating the exponent field of the FP16 scale $\mathbf{S}^{(l)}$ using integer operations, rather than performing an FP16 multiplication.

An FP16 number x is stored with a sign bit, a 5-bit exponent, and a 10-bit mantissa. Its value is

$$x = (-1)^{\text{sign}} \times 2^{\text{exponent\_val}-\text{bias}} \times (1.\text{mantissa\_bits}).$$

The bias for FP16 is 15.

To dequantize a PoT value (sign S_{w}, integer exponent E_w) and combine it with an FP16 scale

$$s = (-1)^{\mathsf{S}_{\mathsf{s}}} \cdot 2^{E_s - 15} \cdot 1.M_s,$$

the dequantized value is:

$$s \cdot (-1)^{\mathsf{S}_{\mathsf{w}}} \cdot 2^{E_w} = (-1)^{\mathsf{S}_{\mathsf{s}} \oplus \mathsf{S}_{\mathsf{w}}} \cdot 2^{(E_s - 15) + E_w} \cdot 1.M_s.$$

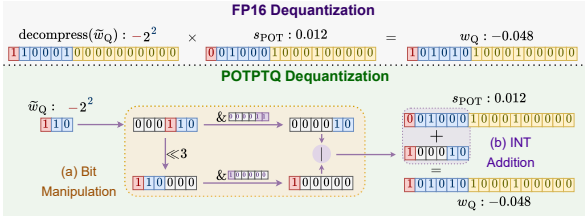

Figure 2: PoT dequantization (3-bit example: 1 sign, 2 exponent bits) for FP16 reconstruction. The quantized sign (S_0) and exponent value (from E_1E_2) are combined with the FP16 scale s by manipulating s's bit representation using integer arithmetic (exponent addition, sign XOR).

This means the PoT dequantization effectively becomes:

1. **Extract PoT components:** From the n-bit storage, extract the sign bit S_{w} and the bits representing E_w. Convert these bits to the integer E_w.
2. **Modify FP16 scale via integer ops:**
   - Treat the 16-bit representation of the scale s as an integer.
   - Add E_w to the 5-bit exponent field of s. This is an integer addition on a sub-field of the 16-bit integer.
   - XOR the sign bit of s with S_{w}.

This process, depicted conceptually in Figure 2, avoids any floating-point multiplication for the $2^{\mathsf{E}_{\mathsf{val}}}$ term. The entire dequantization per element can be implemented with a few bit-wise and integer arithmetic operations, making it extremely fast on GPUs.

5. Experiments and Results

We evaluated POT-PTQ on LLaMA1 (Touvron et al., 2023a) and Llama2 (Touvron et al., 2023b) (7B, 13B, 30B parameters) for 2-bit and 3-bit weight-only PTQ (average 2.25 and 3.25 bits per weight, including sign and shared scales). Group size G was 128. Step 2 calibration used 128 sequences (2048 tokens each) from WikiText-2 (Merity et al., 2016) for 10 epochs (3-bit) or 40 epochs (2-bit). All quantization was performed on a single NVIDIA V100 GPU. Baselines include RTN (Dettmers et al., 2022b), GPTQ (Frantar et al., 2023), AWQ (Lin et al., 2024), and OmniQuant (Shao et al., 2024). Further details are in Appendix A.3.

Perplexity Evaluation. Table 1 details WikiText-2 perplexity (PPL) results. POT-PTQ consistently achieves lower (better) PPL compared to strong uniform PTQ baselines, particularly excelling in the challenging 2.25-bit regime where some alternatives like AWQ falter. Our approach often matches or surpasses even the best-performing uniform methods like OmniQuant at these ultra-low precisions. Critically, naive adaptations of existing PTQ methods for PoT representations yield very poor PPL (see Table 4 in Appendix), confirming the necessity of our specialized two-step optimization for effective PoT quantization.

Table 1: WikiText-2 perplexity (PPL) of 2- and 3-bit quantized LLaMA models. Baselines use uniform quantization; PoT is our POT-PTQ. Lower is better. Our method shows strong performance, especially at very low bit-widths.

Bits	Model	Size	RTN	GPTQ	AWQ	OMNI	PoT
3.25	LLaMA1	7B	7.01	6.55	6.46	6.16	**6.12**
		13B	5.88	5.62	5.51	5.46	**5.42**
		30B	4.87	4.80	4.63	4.58	**4.50**
	LLaMA2	7B	6.66	6.29	6.24	6.21	**6.03**
		13B	5.51	5.42	5.32	5.28	**5.24**
2.25	LLaMA1	7B	1.9e3	44.01	2.6×10^5	**9.77**	9.79
		13B	781.2	15.6	2.8×10^5	**7.93**	7.96
		30B	68.04	10.92	2.4×10^5	7.13	**7.01**
	LLaMA2	7B	4.2e3	36.77	2.2×10^5	11.23	**11.03**
		13B	122.08	28.14	1.2×10^5	8.33	**8.29**

Harness Evaluation. On the Open LLM Leaderboard common sense reasoning tasks (Table 5, Appendix), POT-PTQ generally matches or slightly outperforms OmniQuant on average scores across multiple LLaMA model sizes and bit-widths. This indicates that POT-PTQ effectively preserves not just language modeling fluency but also broader reasoning capabilities.

Ablation Study. To isolate the effect of each step, we perform an ablation study on 2-bit LLaMA1-7B and 13B models (Table 2). Step 1 alone (agnostic initialization) yields reasonable results, demonstrating the effectiveness of our scale grid search. Step 2 alone, without proper initialization, performs poorly due to suboptimal starting points. The best performance is consistently achieved by combining both steps, confirming their complementarity: Step 1 provides a strong initialization, while Step 2 refines it with minimal calibration.

Table 2: Ablation study on LLaMA models (2-bit quantization). PPL on WikiText-2. Step 1: scale initialization; Step 2: calibration fine-tuning. Both steps are crucial for optimal performance.

Model	Step 1 (Init)	Step 2 (Tuning)	Perplexity
LLaMA1 7B	✗	✗	408,838.25
	✓	✗	20,135.70
	✗	✓	51.87
	✓	✓	**9.79**
LLaMA1 13B	✗	✗	40,328.34
	✓	✗	8,267.57
	✗	✓	103.45
	✓	✓	**7.96**

Dequantization Speed. Our custom PoT dequantization kernel, implemented using efficient bitwise and integer operations as described in Section 4, was benchmarked on NVIDIA V100 and RTX 4090 GPUs. Table 3 shows warp cycles for dequantizing a block of weights. Compared to a standard FP16 dequantization for uniformly quantized integers, our PoT kernel achieves a $3.67\times$ speedup on V100 and $1.63\times$ on RTX 4090. This demonstrates the significant inference-time advantage of our hardware-friendly PoT approach.

Table 3: Dequantization efficiency (warp cycles per block). PoT (Ours) is significantly faster due to integer-based operations. Speedup: $3.67\times$ on V100, $1.63\times$ on RTX 4090.

GPU	Arch.	Uniform (FP16)	PoT (Ours)
Tesla V100	Volta	110	30 ($3.67\times$)
RTX 4090	Ada	98	60 ($1.63\times$)

The entire POT-PTQ quantization process for LLaMA1-7B (both steps) completes in approximately 0.71 hours on a single V100 GPU, highlighting its practicality. The fine-tuning efficiency of Step 2 is further detailed in Table 6 (Appendix), showing rapid convergence.

6. Conclusion

POT-PTQ introduces an effective two-step post-training quantization framework for Power-of-Two (PoT) representations in LLMs. It first derives robust, data-agnostic scales via grid search, followed by lightweight, data-driven calibration. This design enables state-of-the-art accuracy at ultra-low bit-widths (2–3 bits), outperforming uniform

baselines in perplexity and downstream tasks. Additionally, our integer-only PoT dequantization kernel offers substantial speedups (up to 3.67× on V100, 1.63× on RTX 4090). Combining high accuracy and efficiency, POT-PTQ is a practical solution for deploying LLMs in resource-constrained settings.

Impact Statement

This work aims to make large language models more computationally efficient and accessible, potentially broadening their beneficial applications while reducing energy consumption associated with their deployment. We do not foresee direct negative societal impacts arising from the quantization methodology itself. Ethical considerations related to LLM applications generally remain relevant.

References

Bengio, Y., Léonard, N., and Courville, A. C. Estimating or propagating gradients through stochastic neurons for conditional computation. *CoRR*, abs/1308.3432, 2013.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

Dettmers, T., Lewis, M., Belkada, Y., and Zettlemoyer, L. Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. *Advances in Neural Information Processing Systems*, 35:30318–30332, 2022a.

Dettmers, T., Lewis, M., Belkada, Y., and Zettlemoyer, L. Gpt3.int8(): 8-bit matrix multiplication for transformers at scale. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 30318–30332. Curran Associates, Inc., 2022b. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/c3ba4962c05c49636d4c6206a97e9c8a-Paper-Conference.pdf.

Elhoushi, M., Chen, Z., Shafiq, F., Tian, Y. H., and Li, J. Y. Deepshift: Towards multiplication-less neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2359–2368, 2021.

Frantar, E., Ashkboos, S., Hoefler, T., and Alistarh, D. OPTQ: accurate quantization for generative pre-trained transformers. In *ICLR*. OpenReview.net, 2023.

Li, X., Liu, B., Yu, Y., Liu, W., Xu, C., and Partovi Nia, V. S3: Sign-sparse-shift reparametrization for effective training of low-bit shift networks. *Advances in Neural Information Processing Systems*, 34:14555–14566, 2021.

Li, X., Liu, B., Yang, R. H., Courville, V., Xing, C., and Nia, V. P. Denseshift: Towards accurate and efficient low-bit power-of-two quantization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 17010–17020, 2023.

Li, Y., Dong, X., and Wang, W. Additive powers-of-two quantization: An efficient non-uniform discretization for neural networks. In *ICLR*. OpenReview.net, 2020.

Lin, J., Tang, J., Tang, H., Yang, S., Chen, W.-M., Wang, W.-C., Xiao, G., Dang, X., Gan, C., and Han, S. Awq: Activation-aware weight quantization for llm compression and acceleration. In *MLSys*, 2024.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. Pointer sentinel mixture models, 2016.

Nagel, M., Fournarakis, M., Amjad, R. A., Bondarenko, Y., van Baalen, M., and Blankevoort, T. A white paper on neural network quantization. *arXiv preprint arXiv:2106.08295*, jun 2021. doi: 10.48550/arXiv.2106.08295. URL https://arxiv.org/abs/2106.08295.v1.

Shao, W., Chen, M., Zhang, Z., Xu, P., Zhao, L., Li, Z., Zhang, K., Gao, P., Qiao, Y., and Luo, P. Omniquant: Omnidirectionally calibrated quantization for large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=8Wuvhh0LYW.

Team, G., Anil, R., Borgeaud, S., Wu, Y., Alayrac, J.-B., Yu, J., Soricut, R., Schalkwyk, J., Dai, A. M., Hauth, A., et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023a.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023b.

Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., Wei, H., Lin, H., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Lin, J., Dang, K., Lu, K., Bao, K., Yang, K., Yu, L., Li, M., Xue, M., Zhang, P., Zhu, Q., Men, R., Lin, R., Li, T., Xia, T., Ren, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Wan, Y., Liu, Y., Cui, Z., Zhang, Z., and Qiu, Z. Qwen2.5 technical report. *CoRR*, abs/2412.15115, 2024.

You, H., Chen, X., Zhang, Y., Li, C., Li, S., Liu, Z., Wang, Z., and Lin, Y. Shiftaddnet: A hardware-inspired deep network. *Advances in Neural Information Processing Systems*, 33:2771–2783, 2020.

Zhang, S., Roller, S., Goyal, N., Artetxe, M., Chen, M., Chen, S., Dewan, C., Diab, M., Li, X., Lin, X. V., et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

A. Appendix

A.1. Additional Figures

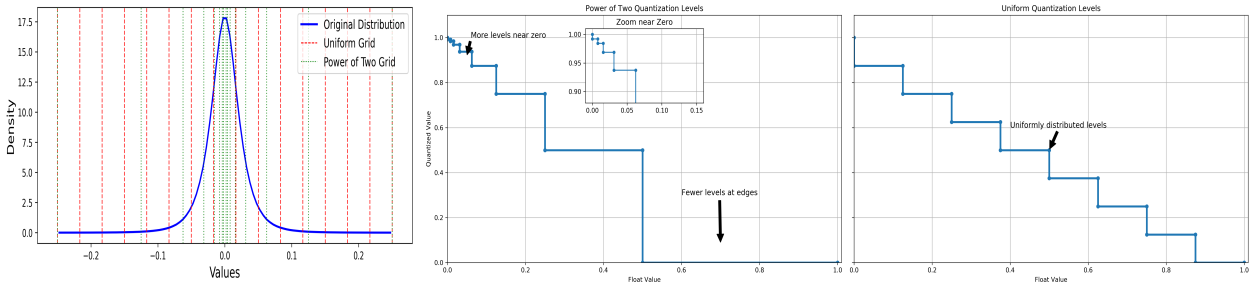

Figure 3: Left: Density distribution of the first weight matrix in LLaMA-7B, exemplifying the bell-shaped or exponential decay commonly observed in LLM weight distributions. Middle: Quantization levels for power-of-two (PoT) quantization, showing finer resolution near zero, aligning well with the distribution's high-density region. Right: Quantization levels for uniform quantization, which allocate levels evenly and poorly capture the dense region near zero.

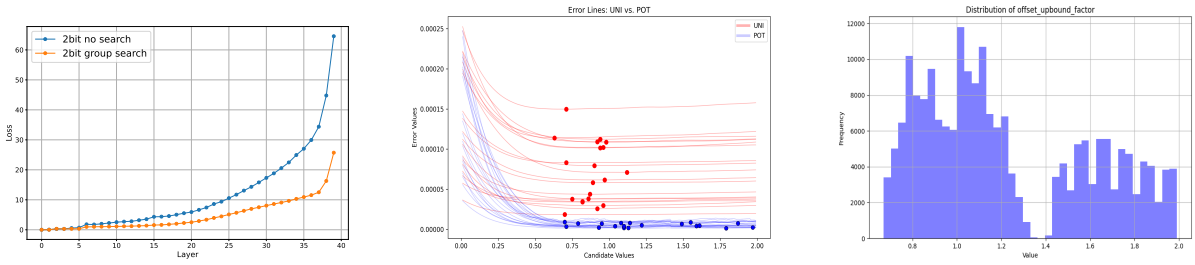

Figure 4: Left: Illustrative loss comparison for Step 1, showing the benefit of careful scale initialization versus a naive approach. (Original caption: "Loss in Step1 O/W Step 1") Middle: Loss curve of 3-bit PoT quantization as a function of the scale multiplier b, showing non-smooth transitions due to discrete exponent rounding. Right: Histogram of optimal scaling multipliers b^* found during Step 1. Many deviate significantly from the naive choice $b = 1$, underscoring the importance of the grid search.

A.2. Algorithms

Algorithm 1 Parallel Data-Agnostic Scale Initialization (Step 1)

for each weight group $\mathbf{W}_{\text{group}}$ **in parallel do**
  $s_0 \leftarrow \dfrac{\max|\mathbf{W}_{\text{group}}|}{2^{q_{\max}}-1}$ ▷ Initialize base scale
  $\mathcal{B} \leftarrow \{0.01 \cdot i \mid i = 1, \ldots, 200\}$ ▷ Set candidate multipliers
  $Q_1^{\min} \leftarrow \infty$ ▷ Initialize minimum error
  for each $b \in \mathcal{B}$ **do**
    $s_b \leftarrow s_0 \cdot b$ ▷ Compute candidate scale
    $\mathsf{E}(b) \leftarrow \text{clamp}\left(\text{round}\left(\log_2\left(|\mathbf{W}_{\text{group}}|/s_b\right)\right), 0, q_{\max}\right)$
    $\widetilde{\mathbf{W}}_{\text{group}}(b) \leftarrow s_b \cdot \text{sign}(\mathbf{W}_{\text{group}}) \circ 2^{\mathsf{E}(b)}$
    $Q_1(b) \leftarrow \left\|\mathbf{W}_{\text{group}} - \widetilde{\mathbf{W}}_{\text{group}}(b)\right\|_2^2$
    if $Q_1(b) < Q_1^{\min}$ **then**
      $Q_1^{\min} \leftarrow Q_1(b)$
      $b^* \leftarrow b$
    end if
  end for
  $s^* \leftarrow s_0 \cdot b^*$ ▷ Optimal scale for this group
  Store s^* into the corresponding position in $\mathbf{S}^{(l)}$
end for
return $\mathbf{S}^{(l)}$

Algorithm 2 Fine-Tuning the Learnable Parameter $\mathbf{\Gamma}$ for Scaling Factors (Step 2)

Initialize: $\mathbf{\Gamma} \leftarrow \mathbf{0}$ (with same group structure as scales)
Set Parameters: Learning rate η, weight decay λ, epochs N
for epoch $= 1$ to N **do**
  for each calibration batch $\mathbf{X}$ **do**
    Compute original output: $\mathcal{H}_{\text{orig}} \leftarrow \mathcal{F}^{(l)}(\mathbf{W}^{(l)}, \mathbf{X})$
    Update scales: $\hat{\mathbf{S}}^{(l)} \leftarrow \mathbf{S}^{(l)} \circ (\mathbf{1}\mathbf{1}^\top + \mathbf{\Gamma})$ (element-wise, respecting group structure for $\mathbf{\Gamma}$)
    Quantize exponent using $\hat{\mathbf{S}}^{(l)}$:
      $\mathsf{E}^{(l)}(\mathbf{\Gamma}) \leftarrow \text{clamp}(\text{round}(\log_2(|\mathbf{W}^{(l)}|/\hat{\mathbf{S}}^{(l)}(\mathbf{\Gamma}))), 0, q_{\max})$
    Dequantize using $\hat{\mathbf{S}}^{(l)}$ and $\mathsf{E}^{(l)}(\mathbf{\Gamma})$: $\widetilde{\mathbf{W}}^{(l)}(\mathbf{\Gamma}) \leftarrow \hat{\mathbf{S}}^{(l)}(\mathbf{\Gamma}) \circ \mathbf{P} \circ 2^{\mathsf{E}^{(l)}(\mathbf{\Gamma})}$
    Compute quantized output: $\mathcal{H}_{\text{quant}} \leftarrow \mathcal{F}^{(l)}(\widetilde{\mathbf{W}}^{(l)}(\mathbf{\Gamma}), \mathbf{X})$
    Compute loss: $Q_2(\mathbf{\Gamma}) \leftarrow \|\mathcal{H}_{\text{orig}} - \mathcal{H}_{\text{quant}}\|_F^2 + \frac{\lambda}{2}\|\mathbf{\Gamma}\|_F^2$
    Update $\mathbf{\Gamma}$ via gradient descent (using STE for $\mathsf{E}^{(l)}(\mathbf{\Gamma})$)
  end for
end for
Return: Refined scale matrix $\hat{\mathbf{S}}^{(l)} = \mathbf{S}^{(l)} \circ (\mathbf{1}\mathbf{1}^\top + \mathbf{\Gamma})$

A.3. Experimental Setup Details

We evaluate POT-PTQ on `LLaMA1` (Touvron et al., 2023a) and `Llama2` (Touvron et al., 2023b) models with 7B, 13B, and 30B parameters. The method targets ultra-low precision quantization at 2 and 3-bit levels (specifically, average bits of 2.25 and 3.25, which includes one sign bit and 1.25 or 2.25 exponent bits on average due to grouping and scale storage). All quantization and calibration procedures are performed on a single Tesla V100 GPU (32GB), while kernel benchmarks are additionally run on an RTX 4090 to assess inference efficiency.

Step 1 (Data-Agnostic Initialization): A grid search over the interval $[0.01, 2.00]$ with step size 0.01 is performed for the scale multiplier b for each quantization group (group size = 128) to identify the optimal initial scale $s^* = s_0 \cdot b^*$. **Step 2 (Data-Dependent Fine-Tuning):** Using 128 randomly sampled 2048-token sequences from WikiText-2 (Merity et al.,

2016), we apply light fine-tuning. The Adam optimizer is used with a learning rate set to 1×10^{-3}, and weight decay for $\mathbf{\Gamma}$ is 1×10^{-1}. Fine-tuning is run for 10 epochs for 3-bit quantization and 40 epochs for 2-bit quantization.

A.4. Additional Experimental Results

Table 4: Perplexity of baseline PTQ methods adapted to PoT format versus our PoT method (POT-PTQ). Naive adaptations degrade performance significantly, demonstrating the importance of a PoT-specific optimization strategy.

Avg Bits	Model	Size	AWQ_POT	GPTQ_POT	OMNI_POT	POT-PTQ
3.125	LLaMA1	7B	6.52	8.27×10^4	6.37	**6.25**
		13B	5.61	5.85×10^4	5.60	**5.50**
		30B	4.72	2.61×10^4	4.75	**4.58**
	Llama2	7B	6.49	NaN	6.46	**6.22**
		13B	5.43	6.41×10^4	5.45	**5.34**
2.125	LLaMA1	7B	2.69×10^5	2.92×10^5	888	**10.86**
		13B	2.80×10^5	1.83×10^5	487	**8.54**
		30B	2.39×10^5	1.44×10^5	297	**7.47**
	Llama2	7B	2.24×10^5	2.78×10^5	3730	**12.80**
		13B	1.27×10^5	1.03×10^5	812	**9.18**

Table 5: Harness evaluation on six common sense reasoning tasks comparing PoT (POT-PTQ) and OmniQuant under 3.25-bit and 2.25-bit settings. Scores are accuracies (%). Bold indicates better performance. POT-PTQ generally maintains or improves performance, indicating robust preservation of model capabilities.

Model	Method	Avg Bits = 3.25							Avg Bits = 2.25						
		arc-c	arc-e	boolq	hs	piqa	wg	Avg	arc-c	arc-e	boolq	hs	piqa	wg	Avg
LLaMA1 7B	Omni	35.6	**64.8**	**71.1**	53.9	77.2	64.5	61.2	26.7	**52.1**	62.2	**40.7**	67.2	55.5	50.7
	PoT	**35.8**	64.1	70.9	**54.3**	**77.5**	**65.2**	**61.3**	**28.1**	50.1	**64.4**	40.1	**67.9**	**57.3**	**51.3**
LLaMA1 13B	Omni	39.8	**72.7**	**67.0**	56.8	77.2	68.7	63.7	**31.3**	**60.1**	63.1	**46.1**	**72.0**	61.8	55.7
	PoT	**40.7**	71.8	65.6	**57.0**	**78.8**	**70.3**	**64.0**	30.1	59.5	**66.1**	45.6	70.3	**62.7**	**55.7**
LLaMA1 30B	Omni	46.0	74.1	71.2	**61.3**	79.6	74.1	67.7	32.4	**65.6**	66.1	49.9	72.3	**62.9**	58.2
	PoT	**47.2**	**73.7**	**71.6**	60.8	**80.1**	**75.0**	**68.1**	**33.8**	64.9	**66.8**	**50.2**	**73.1**	62.5	**58.5**
Llama2 7B	Omni	37.3	**67.6**	71.2	54.5	**76.5**	65.7	62.1	**26.0**	45.0	61.2	39.4	64.5	**54.4**	48.4
	PoT	**38.1**	66.3	**72.0**	**55.2**	76.3	**66.6**	**62.4**	25.8	**52.1**	**63.8**	**40.3**	**65.2**	54.2	**50.2**
Llama2 13B	Omni	41.9	**72.3**	69.9	57.8	78.0	67.7	64.6	**30.0**	**57.0**	63.7	**44.5**	68.0	53.1	52.7
	PoT	**42.5**	70.7	**70.1**	**58.0**	**79.2**	**68.3**	**64.8**	29.4	56.9	**68.2**	43.4	68.8	**56.9**	**53.9**

Table 6: Epoch-wise perplexity (PPL) and loss on WikiText-2 calibration data for LLaMA1-13B during Step 2 of POT-PTQ. Our output-aligned fine-tuning objective consistently reduces perplexity and loss with only 128 calibration samples, demonstrating the efficiency of Step 2.

Epoch	3-bit PPL	3-bit Loss	2-bit PPL	2-bit Loss
1	6.85	6.21	1.01×10^6	25.75
2	5.93	3.86	3209.49	20.45
3	5.70	3.02	239.27	17.13
4	5.59	2.54	94.25	15.39
5	5.54	2.27	49.95	13.50
6	5.51	2.13	31.46	11.99
7	5.49	2.04	23.30	10.55
8	5.48	1.97	18.27	9.60
9	5.48	1.93	15.31	8.65
10	**5.48**	**1.90**	**12.90**	**7.89**